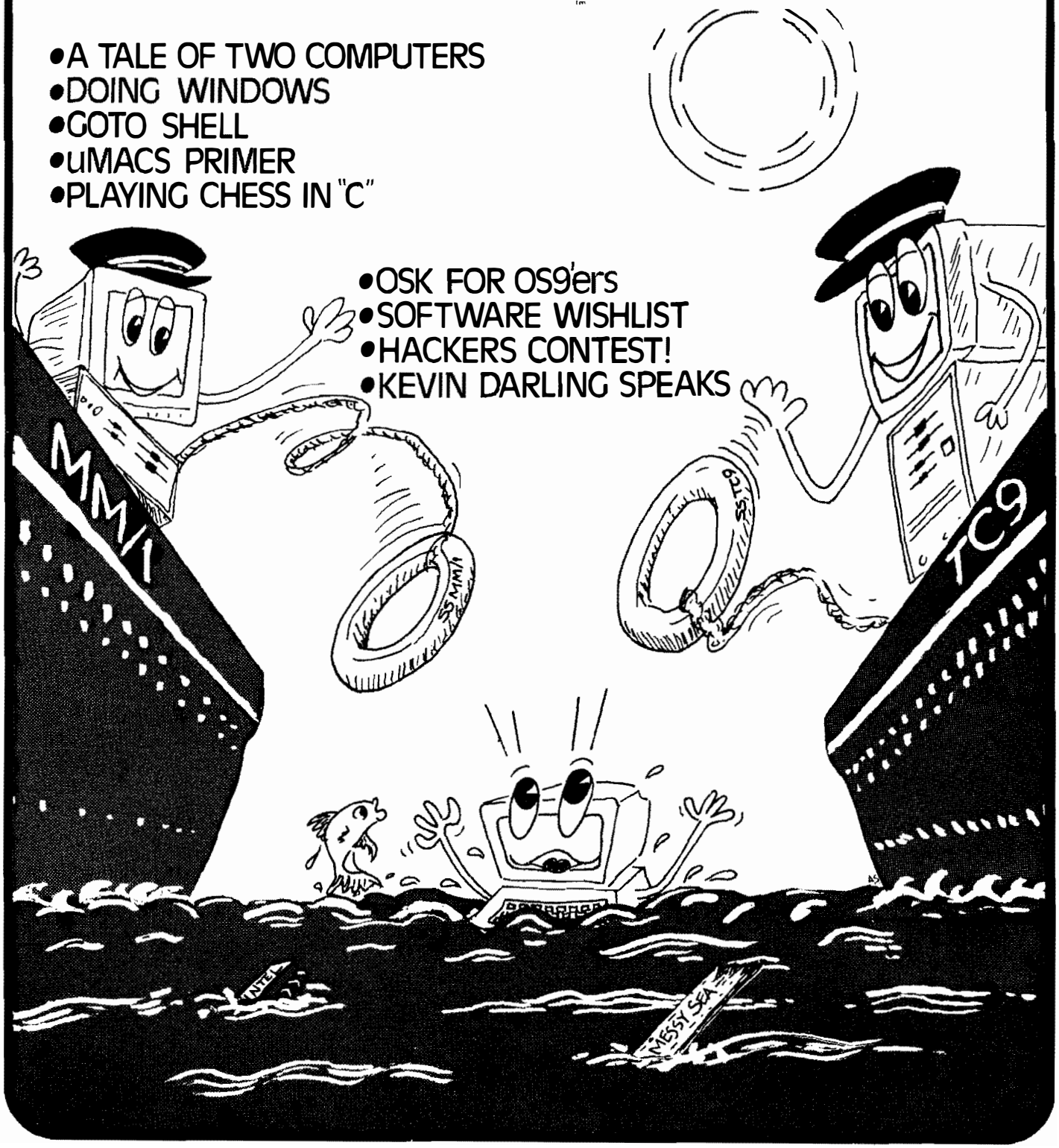
Premiere Issue July 1990 $2.00

THE OSK'ER®

News and Views in the World of OS9/68000 and 6809

- A TALE OF TWO COMPUTERS
- DOING WINDOWS
- GOTO SHELL
- uMACS PRIMER
- PLAYING CHESS IN "C"

- OSK FOR OS9'ers
- SOFTWARE WISHLIST
- HACKERS CONTEST!
- KEVIN DARLING SPEAKS



Directory of /dd/OSKer/Jul90 10:02:46

Owner	Last modified	Attributes	Sector	Bytecount	Name
Editor	90/07/09 0935	----r-wr	3	11550	A_Tale_Of_Two_Computers
OSKer	90/07/09 0756	----r-wr	5	9204	Doctor_OSKer
Bug	90/07/09 0446	----r-wr	7	22826	Doing_Windows
Editor	90/07/09 0757	----r-wr	9	3127	Editors_Ramblings
Editor	90/07/09 0757	----r-wr	10	5844	Flame_ON
Editor	90/07/09 0757	----r-wr	13	2798	Goto_Shell
Editor	90/07/09 0757	----r-wr	14	3061	Hacker_Contest
Editor	90/07/09 0332	----r-wr	14	41602	Kevin_Darling_Speaks
Editor	90/07/09 0737	----r-wr	20	4524	OSK_for_OS9ers
Editor	90/07/09 0935	----r-wr	21	3398	Origins_of_OSKer
Editor	90/07/09 0641	----r-wr	25	9670	Playing_Chess_in_C
OSKer	90/07/09 0935	----r-wr	22	2386	Software_List
Editor	90/07/09 0959	----r-wr	22	5190	uMacs_Primer

"the OSKer" is published monthly by:

StG Computers inc.
P.O. Box 24285
Speedway IN 46224

President: Scott Griepentrog (Editor)
Vice-Pres: Jim Hutchins
Secretary: Chris Swinefurth
Treasurer: Dave Henk

Cover Art: Alan Sheltra

Subscriptions to the OSKer are $12 per year in the U.S., $15 in Canada, and $20 overseas.

Advertising Rates:

4/4 Page	7.5"w X 10.0"h	$ 100
3/4 Page	7.5"w X 7.5"h	$ 80
1/2 Page	7.5"w X 5.0"h	$ 60
1/2 Page	3.5"w X 10.0"h	$ 60
1/4 Page	3.5"w X 5.0"h	$ 40
1/8 Page	3.5"w X 2.5"h	$ 20

Pre-pay for two issues, get one free. Ad copy must be received before last day of month previous to issue for inclusion.

Editing for the OSKer was done on an Atari Mega 2 ST running OSK, using Umacs. PageMaker on a PC (ugh) was used to lay out the pages. Address labels were printed with DB9 under OSK.

As soon as a good DTP becomes available under OSK, (even if I have to write one myself!), all processing will be done using OSK.

All Submissions must be in the Public Domain to be considered for publication. Persons who are selected for publication will be given the following 6 months of the OSKer for free, in addition to any paid-up subscription.

StG Computers inc., as publisher of the OSKer and having ownership of software, will not directly advertise in it nor will the editor use the OSKer to promote his products.

FILE DESCRIPTOR: A_Tale_Of_Two_Computers
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, OS9 Freak
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS,
StG@hummer.iupui.edu

I don't think anybody will argue that the PC machines are starting to look good to even the most die hard CoCo3/OS9 enthusiast. The cost of a base model XT is actually starting to drop below what it costs to put together a comparable OS9 system. Even without the multi-tasking/user features of OS9, a lot of people are finding it hard to resist using what is really an inferior operating system.

As a result of this, a lot of people have been clammoring for newer hardware to bring OS9 into the 90's. Enter FHL (Frank Hogg Labs) and IMS (Interactive Media Systems, formerly Kenneth-Leigh Enterprises), and their new machines, the Tomcat series and MM1.

Going in alphabetical order, I chatted with both camps in an attempt to get some insight on the backgrounds behind the people behind the machines, in addition to the details about what we can expect from the new machines. I think my english teacher just rolled over...

Frank Hogg, from Frank Hogg Labs (FHL)

Frank Hogg is what most people would call a real character. The first thing that struck me is his habit of speaking his mind. He'll tell you what he thinks - if you wanted to know or not. But he's no stranger to the CoCo, OS9, and OSK markets, having sold hardware and software for all. So one is reminded to listen up - he'll give you good tips from his experience - even if you're going to compete against him.

Frank started his business back in 1976 - at first as a dental lab. He got involved in microcomputers first with the KIM1, a 6802 machine. In '79 he got into the software business, and the following year came out with four packages for the FLEX operatin g system. He was also approached at that time by Ken Caplan to support OS9 - then level 1, full of bugs, and had no software.

In '82 came a big break with Frank's nephew Rich Hogg discovering how to upgrade the new 32K CoCo to 64k just by adding four wires. This allowed the several hundred FLEX packages to work on it, and brought him into the CoCo market. Then at the 1984 Ft Worth Rainbowfest the newly ported Level 1 OS9 was being sold for the CoCo, and Frank showed off his O-Pak utilties (also done by Rich). They used the graphics mode to add lower case characters and more than 32 columns to OS9.

In the same year, Frank started selling his first QT computer, a single board 68008 (8 bit bus equivalent to 68000). He subsequently came out with 68000 and `020 models of the QT. But he ran into a problem with the QT line. Upgrading to a faster model required replacing the entire board, which was costly. In '88, he solved this problem by creating the K-Bus system. Although Frank's idea, it was designed by Mike Smith and Dave Bridger at Hazelwood Computers, who also did the very first port of OSK back in '83 and made the QT motherboards.

The K-Bus is a 16 bit bus, that can address up to 16 meg. Having been around for a few years already, has many cards for it already, including math coprocessors, ram, clock/dma/ printer, SCSI, Floppy, Serial, and a 68030 board. Frank claims that it is catching on to the point that other manufacturers are starting to build K-Bus cards.

And now, finally, to the new Tomcat line. The TC9 looks at first glance like it should be a CoCo3. It is actually a 6803 running at 3mhz with a CoCo3 GIME, AT keyboard, 8-bit sound, and other improved features. It has a CoCo bus, so all CoCo hardware will work with it, and fits into the K-Bus for connecting to the 68k world. The board was designed by Bob Puppo, creator of the PC keyboard adapter for the CoCo.

The TC9 can interact with other K-Bus cards, but it must have a 68k processor (a 68000, '30, or TC70 will do) in the bus to handle requests for it. The TC9 can't directly access the K-Bus, although a 68k processor on the bus can access all of the TC9's m emory, and can be triggered to do so via an interrupt. For example, a memory move can be handled a lot faster as the 68k processor has access to the entire TC9's ram without going through the DAT (address translator that allows the 6809 processor to hand le more than 64k).

Frank also points out that more than one TC9 board can be in the bus at the same time. Just think, a whole bank of CoCo's in one box, with OSK running at the same time! Too bad it can't handle multiple 68k processors...

The other member of the Tomcat family is the TC70. It is a 68070 processor, with VSC graphics chip, ram, and serial ports. Although it resembles the MM1 in many ways, Frank claims that the design had been in the works even before the advent of the MM1. It is an alternate 68k processor for the K-bus, with lots of extra features.

Interestingly enough the TC70 board was also made so that it could used as a replacement for the older QT motherboards, and can be mounted on a floppy drive, handy for industrial applications.

As we finished, Frank related a few stories to me. It seems that Paul Ward (IMS) had an advertisement in the Rainbow for three months straight. Then when he skipped a month, Frank happened to put in an ad for the TC9, and a lot of people (without reading the ad?) thought that the TC9 was from Paul...

Frank also says he's working on having the capability to run RSDOS on the TC9, and is seriously thinking about doing a MAC emulator for the TC70. And he doesn't miss a chance to remind me that his port of OSK is more stable, as he puts it. Lots of extra features, like a dmode command that allows you to just say `dmode /d0 -coco' to read a CoCo format disk.

Before I got off the phone with him (several hours later), he said some things about OS9 in general I thought very important to relay. He spoke about the usefulness of Level 1 OS9, ``The ability was there, but nobody wrote the software... Like having a c ar that can do 500 mph but no roads that fast''. With Level 2 on the CoCo3, things were better, but he said, ``mundane everyday appliations [still] have to be done... Bouncing balls are not going to cut it''.

He stressed that OSK developers should look at the phenominal opportunity before them, because ``the first software [is] going to make all the money, [and] everybody else is going to play catch-up''. It's a wide open market with these new machines, as he pointed out, ``everybody who is interested in OSK has to participate''. Finally he convinced me that ``being first is more important than

everything else... when the heat of competion comes on, anounce a new version and add features''.

Paul Ward, from Interactive Media Systems

Paul got his first CoCo back in '82. He did programming on it, PC's and Mac's. In '84 he picked up OS9 Level 1, and fell in love with it immediately. He had trouble with the Rainbow guide to OS9 though. He thought it allowed too many opportunities for user error and decided to write a new one. In '86 he started ``Start OS9, an enjoyable hands-on guid to OS9 Lv2 on the CoCo3'', which took about a year to complete. The biggest incentive was his feeling that OS9 would be responsible for the longevity of the CoCo. In his book he has an essay on CDI, the future of the CoCo, and interviews with Microware staff.

The MM1 machine started with Kevin Pease, Kevin Darling, and Paul talking about the `CoCo4' back in September of last year. Pease, who did the design for the machine, is a top R&D hardware designer for Rand McNally, and Paul claims he works very fast. ``We had drawings worked up by the end of January, with the specs nailed down by surveys on what people wanted and our own design goals''. Four prototypes were done by April, in time for the Chicago Rainbowfest. ``It was actually a lot of fun because our room was packed full with a lot of computer cases and boards'', said Paul. ``We were soldering and putting together systems the thursday night before the fest''.

Paul related how they had some good demos to start off the weekend with, and Kevin Darling was working back in North Carolina uploading new ones each night. Because he was busy, Kevin couldn't lead the seminar on OS9 he was scheduled for. Dale Puckett took over and invited the three attending hardware people for demos and Q/A. All who attended will not soon forget watching Paul and Frank chide each other about the differences between their machines.

Paul says that ``the system that people get in August will be the 3rd version of the board''. He also talks about already having orders from 3 universities for music, multi-media, and porting stuff from Unix. ``We're very agressive about getting software on the MM1", says Paul. ``The only way to survive for ten years is to come out with better technology''. Pease actually suggested the chips selected for use in the MM1 - the 68070 and the VSC, both used in CDI player designs. ``We designed the system to be very affordable by making the cpu board as minimal a computer as possible while still keeping it really sexy.''

Paul also talks about being very commited to OSK and found that Microware was ``very profesional and cooperative''. He's going after the big fish, trying to compete with the PC, Mac, and Amiga because ``that's where the true competition lies''. His plans include ``coming up with mainstream software, permiting mainstream hardware add-ons and still doing something different and better than the other systems''. He also has plans to go after university students.

Rather boldly Paul told me, ``We are going national with this computer (and our next computer) - whoose details are completely under wraps''. I asked what the story was with the MM1's bus. He says that people fear that there will be no more add-on cards. On the drawing board he claims are a digitizer, network (ethernet), serial i/o, and tape backup. It is a 32 bit bus, derived from VME to make adaptation of VME cards to the MM1 bus very easy. He even talks about being able to use PC 16-bit add-on cards in the future.

Finally, I asked him to define Multi-Media, which forms part of the name of his new machine. It simply means that the computer has interactive text, graphics, animation, and sound capabilities. I can just see an MM1 being used as a watchdog - listening for unfamiliar sounds and animating a dog jumping at the door while barking loudly...

So, to sum up...

We've got two new machines. Most people would say think that these two guys are playing a game of dueling computers, trying to outperform each other. It's been very interesting so far this year, and it's only going to get better. But I don't see this as a war at all. It's natural for a little friction to heat things up when two very determined people have two very different ideas about how to do things. What is really happening is that these two people have (possibly without realizing it) given us a choice.

We don't have the much awaited for `CoCo4' machine, the fabled machine that would solve all our problems, we actually have two real machines that give us totally different options, run the same operating system, and will even have compatible windows! We may be sitting back and shaking our heads at these two guys thinking, why? Why couldn't they get together instead of battling it out! But realize that we, the OS9 public, are benefiting from this. Let's just hope they can keep it up eh? Wish them both good luck next time you talk to them, and no matter which one you decide to go with, Keep On OSK'ing!

FILE DESCRIPTOR: Doctor_OSKer
OWNER: John Doe
ATTRIBUTES: Question Answerer, Identity Unknown
ALLOCATION MAP: none as yet

In this regular section, our mythical Doctor OSKer will answer any and all questions about Life, the Universe, and Everything about OS9. Of course, don't expect reasonable answers on the first two. But we are assembling a panel of experts to field those really oddball questions, as well as the just plain ordinary ones.

If you have a question or wish to be on our panel of experts, you can write, call, e-mail, etc., any of the addresses below:

Doctor OSKer
P.O. Box 24285
Speedway IN 46224

(317) 241-6401
Sysop@ROOT (StG-Net); 72427,335 (CIS); StG@hummer.iupui.edu

Now that that's out of the way, let's get on with the questions... What? No questions? Oh, I forgot, this is the first issue. Nobody knows I exist. Let's see, if nobody knows that I exist, does that mean that I don't? Like that tree that can't fall because nobody is there to hear it. Or is my existance relative to my own perception of whether I exist or not...

Anyways, before the good Doctor imagines he's god and takes over the world, let's replace him with some Q's and A's about the new machines. Large portions of this have been derived from Frank Hogg's question and answer sheet (Thanks Frank!), plus conversations with both camps and other unassorted sources.

Q: Will the TC9 be compatible with the MM1?

A: Not really. This is like asking if a CoCo (6809) and a 68K processor are compatible. The TC9 has a 6809 compatible CPU, and a CoCo3 GIME chip, so it's graphics capabilities are the same as the CoCo. The MM1 uses a VSC chip for graphics, which has more resolution and colors. But then the TC9's brother, the TC70, also has a VSC chip and uses the same CPU as the MM1. Comparing the TC9 and the MM1 is like comparing apples and oranges - but the TC70 is close enough to the MM1 to be called compatible, just like the TC9 is close enough to a CoCo to be called compatible. But it is also interesting to note that the TC9 and the TC70 can exist in the same bus. Even though they are not directly compatible, then can share and talk between themselves, thus bridging the gap between 6809 and 68k. The MM1 will also have a CoCo `gateway', which can plug into existing CoCo.

Q: What is the difference between the MM1 and the TC70?

A: The biggest difference is that the TC70 is a card for the K-bus, allowing it to use other cards for interfacing to the real world, whereas the MM1 has a lot of built in features between its two boards. The MM1 is expected to cost less than a comparable TC70 setup, but it is not upgradable as yet. The base MM1 (both boards) has 1M ram, 3 serial ports, Stereo sound In/Out, CoCo compatible joystick, hi-res mouse inputs, and an XT keyboard, whereas the TC70 has 1.5M ram, 2 serial ports, Mono sound In/Out, and an AT keyboard. Otherwise they are practically the same, with the exception that the MM1 will run faster when expanded to 3M ram.

Q: Will I be able to just unplug the Multi-pak from my CoCo 3, with Disto II floppy controller, Burke & Burke HD interface & RS232 pak and just plug the works into the TC9?

A: Yes, because the TC9 has a CoCo Bus everything will work except ROM cartridge games. You may not need the RS-232 pak though because the TC9 has 2 RS232 style ports on it.

Q: Do I still need the multi-pak, or can I connect my existing drives some other way?

A: The TC9 should run 2 Paks on just a cable. The CoCo Bus on the TC9 is via a header rather than a card edge connector. This was done to make it easier to cable the paks in the case. We also put 12 volts back on the bus for things like the Burke & Burke Interface.

Q: Can I mount my floppies and hard drives in the TC9 case?

A: Yes, and there is a 200 watt power supply to handle it.

Q: Do the new computers have a built in mouse interface?

A: The TC9 and the MM1 have a 8 bit joystick port, which can be used with the Tandy joysticks and mice. Both machines can use a serial port for a PC type mouse that has better resolution.

Q: Will the Tandy Hi-Res interface work with the TC9?

A: No, it doesn't have a cassette port. But the MM1 has a built in hi-res interface.

Q: What software is included with the TC9?

A: All the details are not available yet, but it should have a version of Tandy's OS9, modified to work with the hardware differences.

Q: What software is included with the MM1?

A: OS9/68000 V2.3, C Compiler, Basic, graphics editor, text editor, tape backup sw, print spooler, PC file manager, and other to be announced.

Q: What about MSDOS compatibility?

A: Emulating an MSDOS environment on a 68k processor would be too slow, and adding a `x86 processor would probably turn out to be more expensive than purchasing a clone and using it as a terminal to OSK when not running MSDOS. Paul says the MM1 is ``MSDOS friendly'', as it can use DOS disks and will have many MSDOS applications ported to it.

Q: What about Mac compatibility?

A: The Mac uses 68k and hardware that is very similar to the TC70 and MM1. With the addition of a Macintosh ROM set and some interfacing (like that currently available for the Atari ST), both machines should be able to run Mac software. Frank is known to be looking at this closely.

Q: Is the TC9 completely CoCo compatible? Will RS BASIC software work with the TC9?

A: Frank is working on RS-BASIC compatibility, but warns that it may be a while before it is ready and fully debugged. Because of a number of hardware differences, some poorly written or highly protected progarms may not work at all. However, all OS9 software will run on it.

Q: What about power on the CoCo bus of the TC9?

A: The TC9 can handle more power than the CoCo3 could, because it feeds on a 200W power supply. This allows any amount of power consumption from the CoCo bus. Also, 12 volts has been wired, which is needed for the Burke and Burke interface.

Q: How about streaming tape backup?

A: The K-bus 68k processors have it available through the SCSI interface. The MM1's tape backup has yet to be decided.

Q: If I have a hard drive on the TC9 will the tape backup system for 68K back it up?

A: Yes, in theory at least. You would need to have OS9/68K running on a 68k procesor in the K-bus, but it could work. Software will have to be written for this to work.

Q: Will my (Disto, Hemphill etc etc) 512K upgrade work in the TC9?

A: Yes, both plug-in upgrades and plug-in chips can be used.

Q: Do I need OS9/68K to make use of the 68000 with the TC9?

A: No, the 68000 CPU is used by OS9/LII as a speed up device besides being used for OSK. You can get faster LII without OSK by just having a 68000 CPU.

Q: How is the 1 meg Disto upgrade installed in the TC9?

A: Just plug it in, no soldering required. We provided a header for this.

Q: Can I use the new style keyboards that have built in trackballs with the TC9?

A: Yes, the trackball would be connected to a serial port and used like a serial mouse.

Q: Can I use my (CM8, Magnavox) monitor with the new computers?

A: Yes, both the TC9/70 and MM1 support existing RGB analog monitors.

Q: Can I use K-Bus cards without a 68k card? Wil the TC9 work on the K-Bus without a CPU on the bus?

A: No. The TC9 cannot directly access anything on the K-Bus. It has to ask the 68k processor to do its work for it. The TC9's memory (CoCo memory) is the only thing that the 68k sees. They use an interrupt protocol to talk to each other.

Q: Can I run the TC70 without the TC9?

A: Yes, the TC70 is a fully functional 68K color graphics computer with 68K etc etc.

Q: Will the TC70 or MM1 run OS9/LII software?

A: Yes. A 6809 emulator program exists that will run OS9 modules. It may not run them much faster than on the CoCo though, possibly even slower. And there will always be those few programs that don't quite work right with it.

Q: Can the TC9 use more than 1 CoCo cartridge at a time?

A: The CoCo bus on the TC9 is just like the CoCo with the same restrictions. You can use 2 or more with a Y cable like the CoCo, or you can use one of the multi-pak like devices sold for the CoCo (Howard and Orion should work OK)

Q: Will the TC9 autoboot OS9?

A: Yes.

Q: Is the K-bus only 16 bits? Are there any plans to upgrade it to 32 bits?

A: The K-Bus is 16 bits data and 16 meg memory map. The 68000 and '70 are also only 16 bit, so the only reason to change it would be for the '30 and '40. It has been found, however, that the drop in speed by putting a '30 on a 16 bit bus is not substantial - unless you're doing nothing but 32 bit data moves.

Q: How about 1.2/1.4 Meg floppies?

A: The TC70's floppy controller supports all densities including 1.2/1.4 Meg. The MM1 comes with a 1.4 Meg floppy.

Q: I don't have a hard drive now, would it be better for me to get a hard drive that is SCSI compatible for future use with 68K?

A: Yes, although all hard drive systems for the CoCo will work with the TC9, a SCSI hard drive would work better. The MM1 has a SCSI interface, so trying to use an existing CoCo hard drive (unless SCSI) would require a SCSI controller, and would work slower than a new drive.

FILE DESCRIPTOR: Doing_Windows
OWNER: Chris `The Bug' Swinefurth
ATTRIBUTES: Student, Human, Hacker
ALLOCATION MAP: bug@root (StG-Net)

Window? What is a window? Well, most IBM users will say it is something you throw your computer through! As OSK/OS9 users we have the POWER to use windows. Most ``messy'' DOS users do not! We can play chess in one window while downloading the daily news in another and even yet in another window something else and so forth. The possiblities of windows are endless! It doesn't matter what you use computers for, windows will prove addicting! ``But how do you make and use windows?''

In OS9 Level 2 running on the CoCo3 making windows is easy, but in OSK windows have not yet been widely standardized.

We do have a form of windows in OSK, but they don't have all the graphics support as OS9 Level 2 CoCo windows. There is hope - even as I type Kevin Darling is busy hacking away at his keyboard writing new window drivers for OSK! We're all rooting (and waiting) for you Kevin!!!

In the meantime I'll explain OS9 Level 2 CoCo 3 Windows. It is very easy to make a window! Anybody can do it. All you have to do is tell OS9 to attach and set the window. There are two main ways to begin: (1) Attach the window, via the I$attach call in assembly or the Iniz command from shell, to the active window queue, a list of all the active windows. Or, (2) Open a path to the window, via the I$Open call.

After you open the window either by ``inizing'' it or opening a path to it, you make the window custom or use the window data in the device descriptor. If you want to use the data in the device descriptor all you have to do is send the OS9 Select codes to the window. You can do this by typing ``Display 1B 21'' from shell or writing $1B $21 to the window via the OS9 I$Write call. You can also write anything out to the window, but most people just send select.

If you want a custom window you need to send the DWSet OS9 display codes to the window. The parameters for DWSet are as follows:

```
DWSet:= 1B 20 typ lcx lcy szx szy forclr bakclr brdclr
```

Type, called the window type, is the kind of window OS9 will make. Seven different kinds of windows are described in the following list:

Window Type Numbers for OS-9 Level 2 CoCo 3

No.	Cols.	Rows.	Colrs.	Graph. Res.	Mem.
1	40	24	16	-- Text --	2K
2	80	24	16	-- Text --	4K
5	80	24	02	640x192	16K
6	40	24	04	320x192	16K
7	80	24	04	640x192	32K
8	40	24	16	320x192	32K

In addition, one more window type, VDG, exists. Look in upcoming issues of the OSKer for an article explaining in more detail VDG windows. The lcx and lcy args are the X and Y coordinates, in chars, for the upper left corner of the screen. Lcx is the number, from the upper left corner, of chars across, the columns, of the window. Lcy is the number of lines down, rows. The window CANNOT be bigger than the screen. Also, make sure that windows do not overlap. There is a way to overlap windows, but that will be in a future article.

The forclr, bakclr, and brdclr are the colors of the window (foreground, background, and border) the window originally begins with. The last step is to write something out to the window. Usually you will send the Select call ($1B $21) so the window you just created will become the active window.

The Select call is like the CLEAR key - it makes the window that it was written out to the current active window, but only if StdIn is the current active window. For example, if I have a program running on /Term, open up a path to /W1, and I write the Select call on that path, W1 would ``pop'' up on my screen. This way Pop can make the window it just made come up on the users screen - instead of having the user press CLEAR until they got to the new window.

The program `Pop' is a utility that enables you to make a window of any kind you want. `Pop' makes all of its windows full size acording to their window type number. The user tells `Pop' what window type he want and gets the next available window and DWSets and Selects the window. Finally it chains a process (or if no name was specified, a shell) to the window.

First, `Pop' figures out what type of window the user wants. Then, `Pop' opens a path to the next available window, via the wild card window. Opening a path to the wild card window, /W, tells OS9 to get the next window not currently being used. Pop then writes the DWSet call out to the new window. Next, Pop sends the Select call to the window. So, the new window ``pops'' up on the users screen. Pop, then, forks label. Finally Pop chains to the command that the user specified, or Shell if no command was given.

Before `Pop' chains the process it tries to fork a program named `label'. Label is a program that ``steals'' the top line of the window for a label and prints the name of window on it. Label is not required for `Pop' - if it is not found `Pop' will just go on. Label will be printed in an upcoming issue.

Using Pop is easy. The options are all preceded by a dash. The options are: -l -c -#. The -l option tells Pop not to fork Label before chaining. The -c option tells Pop to ``pop'' the current window instead of grabing a new one. Pop does this by DWEnding the window before DWSetting it back. The -# option is the number of the window type the user wants Pop to make. If this is omitted Pop defaults to 2, an 80x24 full size screen. The options can come one right after the other or be preceded by their own dash; Pop does not care. The last thing on the command line is the optional name of another command to chain to besides Shell. If running Pop from a Shell you need to put an `&' after all the arguments. This is because Shell does a wait call after it forks a process and Pop does not return to its parent until the program it chained to dies.

The program `Type' that appeared in the June 1990 issue of the Rainbow page 36 was written by me also, and has a bug. The bug was NOT part of the code I submitted, but inside a modification the Rainbow did to my code prior to printing. The bug is on page 38, in the first and only while loop in the program. The line looks like this:

```
while(-argc > 0 && (*++argv)[0] == `-') {
```

The bug is in the -argc part of the line. It should read:

```
  while(--argc && (*++)argv[0] == `-') {
```

The bug is probably a typesetting error and not the Rainbow's fault. Also, as any experianced programer can tell you the `> 0' part of the code is unnecessary, it will work just as well without it.

Another version of Pop comes with the StG Login package. I rewrote `Pop' to include the -c option, to fix the bug in Pop that only allowed windows W1-W9, and added better command line argument processing. One more version of Pop exists that I have not released, but hope to soon. This version supports the Popping of a VDG window! There are still a few bugs in it, and the size of it made it unprintable, but if anyone wants it drop me a line at root or, after Febuary 1991, at MainLine.

(Editor's note: Chris is reachable as Bug@Root, which is the main node of StG-net. Chris will be putting his own system online soon, at which point he will be reachable as Sysop@MailLine.)

`Pop' is well commented and has no documented errors. Just type it in and assemble it with `asm pop o'. You don't need any special files (except the OS9Defs file) to assemble `Pop', and the source code makes good reading for the best programmer or just a starting novice. After all I always say good source code is better reading than a good book anyway <grin>. You should be able to read the comments and figure out what is going on in the program. Look forward to seeing many programs and articles in OSKer from me, The Bug.

```
* Pop - Make/Change a window and possibly fork a process to it
*       By:   The Bug 1990
*             R.R. 3 Box 321
*             Elwood, IN 46036
*             Bug@Root
*             Voice: 317-552-5707
*
*
*       Public Domain 1990, By: Chris Swinefurth
*
*  USAGE:
*          Pop {-cl#} {command...}
*             -c = remake current window
*             -l = no label
*             -# = window type (1-8) def=2
*           default command is shell
*
*          -c will make Pop destroy and recreate the current window
*          -l will tell Pop NOT to fork Label before chaining to command
*          -# is the window type that the user wants Pop to make or change
*             the current type to. Pop will default to type 2 if not specified
*          Pop will chain to command after making the window and optionaly
*             forking Label. Shell will be forked if no other command is
*             specified. Command MUST come after all other parameters on the
*             command line.

  nam Pop
  ttl Make/Change a window and possibly fork a process to it

  ifp1
  use /dd/defs/OS9Defs
  endc

* module header
  mod modend,modnam,typlng,attrev,start,datsiz
typlng  set   Prgrm+Objct
attrev  set   ReEnt+1

* Data Space
cllcod  rmb   2 this is a wild card varible for os9 display codes
wintyp  rmb   1 this is the varible that holds the window type that pop makes
wnlocx  rmb   1 this is the location of the upper left corner - x cordinate
wnlocy  rmb   1 ``                                           `` - y cordinate
wnsizx  rmb   1 this holds the number of columns the window has
wnsizy  rmb   1 this holds the number of row in the window
forclr  rmb   1 this holds the foreground color of the window - pop uses white
bakclr  rmb   1 ``             `` background color of the window - pop user black
brdclr  rmb   1 ``             `` border color of the window
* Pop uses the last char in the name of the window minus 48 for the border
* color. i.e. window=W2; border color is 2

lblflg  rmb   1 this is a flag that tells pop whether or not to fork label
curwin  rmb   1 this flag tells pop whether or not to pop the current window

cmdadr  rmb   2 this is the address of the command pop will chain to

plymem  rmb 128 this is play mem. that pop uses for the ss.devnam call
stkmem  rmb 255 stack memory

datsiz  equ   .  Size in bytes of the data space; for the module header

modnam  fcs   /Pop/   module name
        fcb   $04      edition number

*  Help message (Pop -?)
hlpmsg  fcc   /Pop {-cl#} {command...}/
        fcb   $0D
        fcc   / -c = change the current window/
        fcb   $0D
```

```
        fcc   / -l = no label/
        fcb   $0D
        fcc   / -# = window type (1-8) def=2/
        fcb   $0D
        fcc   / default command is Shell/
        fcb   $0D
        fcb   $0D
        fcc   /PD 1990 By Chris Swinefurth/
        fcb   $0D
        fcb   $00

* Bad window type message (Pop -3 | Pop -4)
badmsg  fcc   /Pop:nonexistant window type/
        fcb   $0D

defcmd  fcc   /Shell/  Name of the default command
        fcb   $0D
lblcmd  fcc   /Label/  Name of the Label command
        fcb   $0D
defwin  fcc   `/W'     Name of the default window descriptor
        fcb   $00

* Start of Pop
start   clra  clear A
        inca  set A to 1
        sta lblflg pop defaults to forking label
        inca  Set A to 2
        sta wintyp pop has a default window type of 2
        clr curwin clear flag to change the current window; default is off

        bra getarg branch to get first byte of the parameter string

* pop branches past the gnxarg to getarg to avoid addvancing the pointer of X
* if the pointer was advanced and the user gave pop a command then pop would
* chain to the second char on of the command string. i.e. Pop Dir
* would make pop chain to ``ir''; by branching past the lda ,x+ pop avoids this

gnxarg  lda ,x+  just advance X
getarg  lda ,x   load A with the byte pointed to by X

        cmpa #$0D test for end of line
        lbeq ldxcmd if eol there must not be a command so, branch an load shell

        cmpa #$20 test A for space
        beq gnxarg if A is a space get next arg

        cmpa #'- test A for a dash
        bne mainlp if a is not a dash then it MUST be the first char of command

        lda ,x+  load A with char at X and advance X

* I threw away one copy of A, because it contained dash;
* this is so I can get  past the dash an get the option after the dash
nxtarg  lda ,x+

        cmpa #$20 test A for space and if it's space get the byte
        beq getarg
        cmpa #$0D test for end of line
        beq ldxcmd if end of line default command is shell

        cmpa #'? test for `-?'
        beq helpme if user wants help message; give it to them

* I or'ed the byte in A by $20 because that will clear the 6th bit and
* the sixth bit is always set for a number and lower case letter
* this way I won't have to test each char for upper and lower case

        ora #$20 A now has a lower case letter or number in it

        cmpa #'l test for `-l'; -l tells pop not to fork label
```

FILE DESCRIPTOR: Editors_Ramblings
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, OS9 Freak
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS, StG@hummer.iupui.edu

Welcome to the first issue of this new monthly magazine for OSK (OS9/68000) and OS9 users. Because OS9 and OSK are essentially the same operating system, just on different processors, I will tend to use OSK (as it is more powerful) to represent both.

This is a very exciting time for everybody! With new computers and software being produced, the OSK market is about to take a flying leap over some tall expectations - and I think I speak for the majority when I say it's about time. Everyone who I've ever met that has gotten into OSK has gotten behind it. In fact, most OSK'ers (I say `oscars') have a dim view of MSDOS, going as far as comparing it to RSDOS on the CoCo.

Let's face it, we have in OSK the best _designed_ operating system - although not the most supported, or even recognized. It's got all the multi-tasking and multi-user (and now multi-media) capabilities of Unix, which the so-called experts say that the software industry is going to move to eventually, but OSK has several major advantages. It is faster and more effecient (being written in assembly) and has built-in record locking and other real-time processing features. I mean, we already have what the rest of the world is trying to do with the likes of OS/2.

Hey, it's our little secret - but not for long. Comparing OSK to Unix, MSDOS, OS/2 (he-he), etc., one can only conclude that once we have a sufficient software base to attract the average user, OSK is going to take over as the new `standard' simply because it's best. When they where looking for an operating system for CD-I (Compact Disk Interactive), they had some very strict requirements - all of which OSK met easily. I take this as a sign that the inevitable is already happening - and I think that the reason CD-I hasn't taken off much as yet is because of it's high price tag.

So it is up to us - the developers, the midnight hackers, the weekend programmers, the students, and everybody else who believes in OSK to band together to push it to the top. Create new hardware, new software, and a new set of standards for the rest of the industry to gawk at. We have the technology, we have some of the best brains, and we have the operating system. Go forth ye programs and multiply...

And that, my friends, is the reason this

magazine was created.

But it was also created to spread the news to those who have not yet become believers, and also to keep information flowing from the developers to the end users. To spread the latest and greatest happenings in our corner of the universe throughout the known galaxy of computing stardom...

Hey, have I started rambling yet?

p.s. pardon the abundunce of articles written by yours truly, but we gotta start somewhere. Get your articles in (on any subject, OS9 or OSK) and see if you can drown me out eh? See your name up in lights... Well, black ink anyways. Plus, get six months of the OSKer for free if we print your article or program!

FILE DESCRIPTOR: Flame_ON
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, Publicly Executable
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS, StG@hummer.iupui.edu

Somebody got a match?

In this regular section, readers are welcome to flame on (or off) about something they take exception to. To start it off, I present something that really ticks me off. As with all posted smoking allowed areas, all statements are opinions of the individual doing the smoking, and should be taken as such.

Like Unix, we have in OS9 and OSK the capability to tag each stored file with what person is responsible for it. In the case where only one person is using the system, this features doesn't have much use. But if you allow someone else access to your machine, and you want that person to access only certain files and commands, it becomes a very powerful feature indeed. Just ask somebody who runs a login system (BBS) with several hundered users.

When they (Microware) created OSK in the image of OS9, they left the filesystem (the way files are stored on disk) alone. This way, all OS9 systems can read OSK disks, and all OSK systems can read OS9 disks (provided the disk formats are compatible). That was a really good idea to keep compatibility with the old format.

But they did change user numbers in OSK - in the process descriptor that is. The process descriptor contains information controlling every program running in the system (see the command procs). Each process has a user number, which is attached to each file it creates. This

```
        bne arg.c if not -l branch over the clear label flag
        clr lblflg user doesn't want Pop to call label; clear label flag

arg.c   cmpa #'c test for -c; -c tells pop to DWEnd the window and remake it
        bne arg.1 if not -c branch to the next test
        inc curwin set current window flag
* setting curwin tells pop not open a new window, and to DWEnd the current one

arg.1   cmpa #'1 test for -1; pop needs to see if the arg is a number or what
        bcs nxtarg if lower get next arg; this arg is bogus; no error
        cmpa #'8 top boundry for number is 8
        bhi nxtarg if higher get next arg; bogus arg; no error

* since OS9 has no window for types 3 and 4 test for them and print help
* message and exit if user tells pop to make type 3 or 4 window. The program
* Type that appeared in the June 1990 issue of the Rainbow supported type 3&4
* windows; type 3 was a 38 column window and type 4 was a 106 column window
* Type 3 was made by setting wnlocx to 2 instead of zero. Type 4 was made
* by using the small graphics font on a type 7 window.
        cmpa #'3 test for -3
        beq bdtype
        cmpa #'4 test for -4
        beq bdtype

        suba #$30 make ascii number a number value
        sta wintyp user wants a window type other than 2; store it in wintyp
        bra nxtarg get the next arg

* Help Message Routine
helpme  leax hlpmsg,pcr load X with address of the help message
getchr  lda ,x load A with char at X
        lbeq gdexit if A=zero do an exit with no error
* zero marks the end of the help message

* Write Line stops writting when it reaches $0D or the amount of chars to write
* the amount of chars to write is always 80 because no one line of the help
* message is longer than 80 chars
        ldy #$0050 load X with 80; max number of chars to write
        lda #$02 load A with StdErr - perfered over StdOut for help message
        os9 i$writln write a line of the help message
        lbcs bdexit if an error occurs while writing exit with error

* this routine advances past the end of line char
nxtchr  lda ,x+ load A with the char in X
        cmpa #$0D Test A for End Of Line Char
        bne nxtchr if not End Of Line get the next char
        bra getchr do routine over aagain

* this routine prints the message that the user gave type a bad window number
bdtype  leax badmsg,pcr load X with address of bad option message
        ldy #$0050 load Y with 80
        lda #$02 load A with StdErr
        os9 i$writln write the string
        lbcs bdexit if error writting error string; exit with error
        bra helpme print user help message

* this is the DWEnd routine
endwin  ldd #$1B24 load D with DWEnd codes
        std cllcod store D in cllcod varible

        leax ,u load X with address of varibles; cllcod is the first varible
        ldy #$0002 load Y with number of bytes to write
        lda #$01 write DWEnd to StdOut
        os9 i$write with DWEnd to window; this will destroy current window
        rts return to routine that DWSets window

* Main Routine
ldxcmd  leax defcmd,pcr load X with address of the default command; Shell
mainlp  stx cmdadr store the address of the command to chain to in cmdadr
```

```
* If the user wants to ``pop'' the current window don't close StdOut, and
* branch to routine to get window name.
        lda curwin load A with current window flag
        bne getnam if curwin flag is set branch to getnam

* Pop closes StdOut and opens a path to the wild card window. Since OS9
* assignes the lowest path number available when it opens a path StdOut
* is now an open path to the next available window. Later Pop will dup
* StdOut to StdIn and StdErr.
        lda #$01 load A with StdOut
        os9 i$close close StdOut
        lbcs bdexit if error in closing StdOut exit with error
        leax defwin,pcr load X with address of default window desriptor name
        lda #$03 load A with read/write attr's for Open call
        os9 i$open open next available window as StdOut
        lbcs bdexit if error opening window; exit with error

* Pop needs the name of the window so it can make the window border the proper
* color.
getnam  leax plymem,u load X with address of the play memory
        lda #$01 load A with StdOut
        ldb #$0E load B with SS.DevNm code
        os9 i$getstt get the current window name in plymem
        lbcs bdexit if error pulling name exit with error

* Pop usses the last char in the name of the window to determine the border
* color. So, this next routine finds the last char of the name. The last char
* has the high order bit set. Since Pop uses the last char if the window name
* is W1 then the border color will be 1, and if the window name is W27 then the
* border color will be 7 not 27. An intresting thing is some window have less
* than 16 colors, but the border can be any one of the 16 palletes. This allows
* a two color window, such as window type 5 to have 3 colors on the screen.
pulchr  lda ,x+ load A with char at X
        bpl pulchr if char at A isn't negative get next char
        anda #$7F make A a positive
        suba #$30 make char in A a number
        sta brdclr store window number for border color


* If the curwin flag is set then the user wants to pop the current window.
* So, branch to the subroutine that DWEnds the window.
        lda curwin load A with flag for current window
        bsr endwin if current window flag is set; destory current window

* This routine sets the vribles according to the window type the user set.
* Then it makes the window on the path StdOut. If the DWSet routine has been
* called via the curwin flag being set this routine remakes the window.
        ldd #$1B20 load D with DWSet and store codes in cllcod
        std cllcod
        clr wnlocx
        clr wnlocy
        lda #$50 load A with 80 and store 80 in wnsizx
        sta wnsizx
        lda #$18 load A with 24; Pop only supports 24 lines screens
        sta wnsizy
        clr forclr set the foreground color to white
        lda #$02 load A with 2 to set background color to black
        sta bakclr

* These next routines set 40 columns for window types 1,6,and 8
        ldb #$28 load B with 40; some windows use 40 columns instead of 80
        lda wintyp load A with the window type number
        cmpa #$01
        bne cmp.6
        stb wnsizx

cmp.6   cmpa #$06
        bne cmp.8
        stb wnsizx
```

is how you can tell if Joe Hacker created that oddly named file in your cmds directory....

The user number in OS9 is 2 bytes long (0-65535). It is stored that way in the process descriptor, and the same way in the file descriptor segment for each file. In fact, there is only space on disk for the two bytes - it cannot be expanded without departing from the standard set forth by OS9.

The user number in OSK is 4 bytes long. That's nearly half a billion different combinations! Which is an improvement, right? But how do you fit those 4 bytes into the 2 byte space on disk? How do you fit 4,294,967,296 into 65,536? When a user creates a file, how do you keep track of which user created it!

To solve this problem, and help keep track of so many different user numbers, the 4-byte user number format in OSK was split into two 2-byte numbers. The first one is the GROUP number, and the second the USER number. In the password file they are separated by a dot, so that Group 1, User 1 is specified as 1.1.

Okay, this is cool. So you take the USER number and use it to tag files when they are created, but you also allow that user to access files owned by the GROUP number. This way the same old user number format (0-65535) is retained, while adding a pretty darn nice feature. You can have a particular user number which is really a set of files for a group of people to access (like working on the same project). You can add any person to the group at any time - just change their group number. Hey, this is great! A really useful, neat feature!

But that's not what they did.

When converting the 4-byte user number attached to the process into the 2-byte space on disk, they take the lower byte from the GROUP and the USER numbers and stick it together to form the user number on disk. The higher byte of each is thrown away.

Let's take a closer look at this shall we? Group 1, User 1 becomes the user number 257 on disk (and to OS9!). But, Group 1, User 257 also becomes number 257! These two users might as well be the same! The higher byte is dropped - meaning that there are really only 256 distinct user numbers. But the same goes for the Group number. Group 257, User 1 is also converted to number 257. Which means that there are only 256 groups too! Wait a sec, that means we're right back to 256 x 256 = 65536 users! So why the heck did they make this change?

The idea was to add this group number to allow a collection of people access to each other's files to work on the same project. To do it, they changed the way that `public' files work. You can take the public access of your file, and nobody else can touch it. That is, in OS9. In OSK, anybody with your same group number can still access it. Which means that if you want your files private from everybody else you have to be given your own group number. But remember there's only 256 of them!

So we've gone from having 65,536 separate users to only 256. What more could they have mussed up? Well, get this. Group 256, user 256 converts to... 0. That's right, that user not only has access to everything, but you can't tell what he's done! He might just as well be sysop! But is there any warning about this? No, they leave you to discover that major security loophole yourself!

The bottom line? There are still only 65,536 users, but only 256 private file areas. Don't use Group or User numbers above 255, and be careful what users you put in the same group. If you have to move somebody from one group to another, you have to change the owner number of all their files. Watch out for converting user files from OS9, the numbers won't match up.

While some will be quick to point out that few people will really need more than 256 private file areas, this problem still plagues those who do. For example, it is mpossible to OSK use in a college environment where many different student file areas must be kept separate without hacking the system. I really think that was a mistake.

Now I can't agree enough with 99% of what Microware has done with OS9, but this is embarrasing. I really have to wonder at what could possibly move them to make such an obvious break from their previously wonderful system design track record.

Next Time: Proper Formatting of C Source Code

```
cmp.8   cmpa #$08
        bne cmp.5
        stb wnsizx

* Window type 5 only has 2 colors on screen. So, Pop must make the background
* color 1 not 2
cmp.5   cmpa #$05
        bne wrtwin
        dec bakclr

* This routine makes the window via the DWSet call.
wrtwin  leax ,u load X with address of the data space
        ldy #$000A load Y with 10; 10 is the number of bytes to write
        lda #$01 load A with StdOut path
        os9 i$write write the DWSet call to StdOut
        bcs bdexit

* Select the window via the OS9 Select call. So, the window just made will
* ``pop'' up on the user's screen.
        ldd #$1B21 load D with Select codes and store them in cllcod
        std cllcod
        leax ,u load X with address of data space
        ldy #$0002 two bytes to be written
        lda #$01 load A with StdOut
        os9 i$write write the Select codes
        bcs bdexit

* This next routine close the StdIn path and dups the StdOut path. So, StdIn
* not is coming from the newly made window.
        lda #$00 load A with StdIn path, and close StdIn
        os9 I$close
        bcs bdexit
        lda #$01 load A with StdOut, and dup StdOut to StdIn
        os9 i$dup
        bcs bdexit

* This next routine closes StdErr and dups StdOut to StdErr. StdErr will now
* be to the newly made window.
        lda #$02 load A with StdErr, and close StdErr
        os9 i$close
        bcs bdexit
        lda #$01 load A with StdOut, and duplicate StdOut to StdErr
        os9 i$dup
        bcs bdexit

* Test the lblflg and branch over the fork call to chain if clear
        lda lblflg
        beq chain

* This is the routine to fork label
        pshs u push the address of the data space on the stack
        ldy #$0001 load Y with 1; the size of the parameter area
        leax lblcmd,pcr load X with the address of the Label command
        tfr x,u transfer address of command to fork to U register

fndeol  lda ,u load A with char at U
        cmpa #$0D test A for end of line
        beq forkit if A is eol fork label
        lda ,u+ get next char in A from U
        cmpa #$20 test A for space
        bne fndeol if not A space get next char

forkit  ldd #$1100 load D with module attributes
        os9 f$fork fork the label command
        bcs bdexit if error forking label exit with error

* Wait for Label to exit
        puls u get the data space address off the stack
        os9 f$wait wait for Label to exit
        bcs bdexit if an error occurs while waitingl exit with error
```

```
* chain to command
chain   lda curwin load A with current window flag
        bne gdexit if the curwin flag is set user
doesn't want to chain
        leax stkmem,u load X with address of the
stack memory
        tfr x,s put address of stack in stack pointer
        ldy #$00FF load Y with 256
        ldx cmdadr load X with address of the command
        tfr x,u put command address in U

tstchar lda ,u load A with char at U
        cmpa #$0D test A for e o l
        beq chncmd if e o l chain
        lda ,u+ load A with next char
        cmpa #$20
        bne tstchar

chncmd  ldd #$1100 load D with module attributes
        os9 f$chain chain to command

gdexit  clrb clear B for exit without error
bdexit  os9 f$exit exit

        emod  module CRC
modend  equ * bottom of module
```

FILE DESCRIPTOR: Goto_Shell
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, Sea Shell'er from way back
ALLOCATION MAP: Sysop@Root (StG-Net),
72427,335@CIS, StG@hummer.iupui.edu

As luck would have it, OS9 is a little lack in neat shell utilities. Things like the Unix Cshell has - alias, if/then, goto, etc. Hey, we only just got (in OSK) the shell environment variables from Unix, but we still don't have a goto?

Well, as luck would also have it, there's a way to write a goto, in C (or most any other language), and I happen to have done it. Actually, the idea hit me when I was watching the MM1 back at the Chicago Fest earlier this year. Somebody had rigged a neat little hack to allow a shell script to repeat forever - handy for running those fancy graphics displays.

The program was actually nothing more than a lseek() (F$SEEK) call, on path #0 (stdin, therefore the shell script itself) back to the beginning of the file. Place the program at the end of the script, and it rewinds the file to the beginning - and the shell starts receiving the list of command all over again.

The program below, named goto, has a little extra feature. It actually seeks out a particular line in the file you want to jump to. To name a line to jump to, use the * (which is ignored by the shell) and follow it with a word. Then do a goto with that word, and the shell will jump to that point. For example:

```
*repeat
date -m
sleep 60
goto repeat
```

This will display the date and time, wait a second, and repeat. Again, and again. Now I admit that a goto alone has practically no use except for making an endless loop, but I am working on an if (yup, later issue), which will liven things up a bit.

The following code was written and tested on OSK, but should work in OS9 just as well.

```
/* goto (label)
   pd 90/04/21 by StG
*/

#define ERR (-1)
#define BUFSIZ 256

extern int errno;

char find[BUFSIZ];
char buf[BUFSIZ];

main(argc,argv)
char **argv;
{
    if (!*++argv)
    {
        writeln(1,''use: goto (label) - shell script use only\n'',80);
        exit(0);
    }

    /* build string to look for in find */
    strcpy(find,''*'');
    strcat(find,*argv);

    /* rewind script to beginning */
    if (lseek(0,0L,0)==ERR)
    {
        /* should get error if seek is attempted on SCF */
        writeln(2,''GOTO: use in shell script only\n'',80);
        exit(errno);
    }

    /* and look for `find' */
    while (readln(0,buf,BUFSIZ)>0)
    {
        /* continue until we find line that matches */
        if (strncmp(buf,find,strlen(find))) continue;
        if (*(buf+strlen(find))!='\n') continue;

        /* found our label - just exit normally
        and shell will process next line of script */
        exit(0);
    }

    /* didn't find label - display message and exit w/error */
    strcpy(buf,''GOTO: label not found: ``);
    strcat(buf,*argv);
    strcat(buf,''\n'');
    writeln(2,buf,80);
```

FILE DESCRIPTOR: Hacker_Contest
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, Hacker
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS,
StG@hummer.iupui.edu

There are many definitions of the term `hacker'. The one I prefer is `person who spends more time than he should hacking at the keyboard of a computer'. Now there are those that would claim membership in hackerdom by breaking into and crashing computers, or other such frown-upon acts, but these people should be termed malicious hackers, or trashers. Or phreakers if they are fooling with Ma Bell. But I claim to be a `benign' hacker. I could probably get into a system if I really wanted to, but even if I were to I wouldn't do anything to it. Just let the sysop know of the security hole.

So being a true hacker is more a case of being able to figure things out, invent new ways of doing things, etc. Not just happening to know a back door and the command for shutting the system down. Come up with a neat way of doing something (often called a neat `hack'), and you too can call yourself a REAL hacker. And since there's no better challenge than to write a program, I present the following problem.

Write a program that outputs the numbers 1 through 10 in a random order, but using each number once, and only once.

Sounds easy eh? One is tempted to just write a routine that goes through a loop 1 through 10, grabbing random numbers and checking off the ones you have used (so as not to repeat them) as you go. But if you expand that routine to do say, 1000 numbers, you will find that it begins to slow to a crawl as it aproaches the end of the loop (although on an 68k machine you may have to increase that even more to see it). The chances of finding the one number it has left at the end are 1 out of 1000, which means it spends a lot of time looking. So how can it be done that would improve the speed? That is the challenge. Get out your keyboard and start hacking. Write your routine in Basic09, C, or even RS-BASIC. C of course is my favorite, but use whatever you have available.

The best solution each month will be printed, and that person will get a free year's subscription to the OSKer. Also, you can get your picture on the front cover as the `Hacker King'. Submit your code via mail to the OSKer post office box (see inside front cover) or via e-mail to any of my adresses listed above.

Good Luck!

Example of the worst case program:

```
/*
 * rnd(x), returns 0-(x-1) pseudo-randomly
 * this random routine really sucks, but is just as an
example
 * in case your library doesn't have one
*/
rnd(x)
int x;
{
        static long seed=25849;

        seed*=7;
        seed+=3;
        if (seed<0) seed=-seed;
        return(seed%x);
}

/* try setting this to 1000! */
#define MAX 10

/* C happens to zero globally declared stuff */
int array[MAX];

main()
{
        int i,r;

        /* loop to max */
        i=0;
        while (i<MAX)
        {
                /* get rnd's until find one not used */
                do r=rnd(MAX);
                while (array[r]);

                /* print it out and increment use flag */
                printf(``%d\n'',r+1);
                array[r]++;

                i++;
        }
}
```

FILE DESCRIPTOR: Kevin_Darling_Speaks
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, Humble Before The Great Kevin
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS,
StG@hummer.iupui.edu

Kevin Darling, the famous guru of OS9, has graced my humble magazine with an interview. Well, actually, Keven is just pretty darn smart, and is one of the friendliest people you'd ever want to meet. But his name is up there in lights, or should be, and I took this chance to find out why, and what he's up to these days.

As our conversation was recorded on a super high quality Radio Shack answering machine that is not old, just classic, I later had lots of fun transcribing what Keven said out from under the static and warblies. So if you come across something that doesn't sound right, it probably isn't. Words I couldn't decipher at all are marked with [?], and larger holes with [...]. And anything in brackets is essentially what was said, or as close as I could figure, just not the exact wording. Of course, K: is where Kevin has spoken, and S: is where I sneaked in a few words. Enjoy!

K: The biggest thing that's held back OS9 from wide acceptance all these years is not having a graphics standard, or even a terminal standard in the old days.
S: But OS9 Lv2 has its own graphics and windowing standard, so to speak, so you must be talking about OSK?
K: Well, OS9 in general. The cocos were the only ones to have windows for quite some time of course. In Japan, they had windows on their OS9 machines, but they were totally different from ours.
S: How were they different?
K: Well, I've not actually seen them, but I know a guy who was porting a super graphics program called the egg from japanese version to CoCo windows and the project got

cancelled. It's really too bad as it was the most astonishing graphics program you've ever seen in your life. Had incredible manipulations built into it.
S: Sounds like the story of OS9, lots of great stuff written and projects keep getting cancelled. What is it that the MBA types have it against us ya know?
K: <laughs> Politics, politics, politics...
S: Yah.
K: That's the first thing people ask. Oh, OS9 sounds great, but does it have windows? Does it have graphics? You have to say, well, yah and no. There are a half dozen packages out there you can get. There's CoCo windows, Rave, G-windows, MGR which is PO, GKFMAN, and another one or two somewhere. But there's no one standard. And here you are telling everybody [that with OS9] you can take you disks from Macintosh (with compatible drives) to an Atari ST, to an Amiga, and run the same binary program, and hey, this is really cool...
S: Yah! The same program because all those machines use the 68k processor. That is what we need.
K: And if you have a PC drive on the MAC than you can actually the same disk from machine to machine. That's very powerful stuff and what an incredible market that would be, but with no standard graphics, you're [out of luck]. Now you could use a graphics termcap type of deal...
S: But that gets into some complex stuff...
K: Now there are people who have done that. I know a guy who has ported PC programs to OS9 and he uses the same graphics termcap so it will work on anything. Now thats kinda OS9-like in a way...
S: Is it compatible with the termcap from unix?
K: Yah, it's an add-on to that. I think it's something that's already been done under Unix. Now what's funny is that even like three years ago, two years ago there was a huge interest in OS9 building for a while there.
S: From who? Just the CoCo'ers?
K: On CompuServe from people with other machines. Whenever there's an article in magazine about OS9 we get a lot of people come over. In Januaray of 1988 or something there was an article in Dr. Dobbs Journal on OS9. Somewhere around here I got a master list of places to look for OS9 articles. I collect that kind of information.
S: Of course!
K: So when somebody asks me, they've heard of OS9, [I tell them] look in Unix world so and so an issue, the title of it is ``When Unix is the wrong choice''.
S: Great title!
K: And it talks about OS9 being used at NASA. And how for any realtime stuff they only use OS9. Actually, that's kind of a misnomer because they use their own customized version of OS9. They bought a liscense to it.
S: You mean they re-wrote OS9 for their own use?
K: Yah, they modified it for their own use. They call it PCOS.
S: It still runs on 68k's though?
K: Yah
S: What did they change, do you know?
K: I have no idea, I really don't. The shuttle communication system for instance runs off OS9, or that version of OS9. They use the 68010 and they have I think it's 200 channels and 130 intercom positions for all the technichons for when the shuttle takes off and that machine actually takes samples from each of the channels and multiplexes them in real time for people who are punched into each other.
S: Cool! Now thats complex!
K: So anyways, there's been `waves' of interest in OS9 over the years as other systems, like OS/2 was introduced.
S: Yah, like all the people that go ``OS/2's cool, but you've already got OS/9? What the heck [is that]?''
K: But the first question always was that did it have a standard graphics interface, and you have to say no, and interest died again. So we've need a graphics standard for a long, long, long time.
S: And you hope to provide that.
K: And it will be ported to the Amiga, and the Atari.
S: And all the big 68k machines.
K: It may be awful hard to port this to the Mac, because they're running their windows under the Mac operating system. So that's going to be a little bit tougher. [...] I would actually like to use Rave, but they didn't really design [?] it's really for controlled displays. It's not really a multi-screen type of deal. You can't just start a new window like you can on the coco and get a shell on it. They could modify it to do it, and I actually talked to the guys about it before. They're (Microware) interest is still mainly [?] I think still 80% of their business is controlling [systems?..]
S: The industrial market with doing like, process control systems.
K: One of the interesting things is when all of this new [business] with the new machines starting to come out was why don't we just go straight to OS-9000.
S: Now I have never actually seen it, but my opinion of OS-9000 is that it just can't be right because it's not running on a motorola processor.
K: Yah, that's my opinion too. But's it's nice, and it runs, and a 386 processor is a fast [omitted] processor.
S: From what I understand though, OS-9000 only runs on a 386.
K: Yah, a 386.
S: Cuz I had heard rumors that they were trying to do it [OS9] for the PC, and then for the 286, and this comes out and it's 386 only. Like I'm not going to buy a 386 machine just to have OS-9000.
K: Yah, but boy I tell you what, wouldn't it be sweet, cuz you can go out and buy a 386 portable now, a laptop. Can you imagine running OS-9000, that would be great.
S: Well, you could also take an MM1 motherboard and create you own portable laptop out of it.
K: True, but look at all this cheap hardware laying around.
S: That's the problem, yah. They've already got the hardware there.
K: Yah, and that was the main point. Not only that but you automatically have an MSDOS compatible machine at the same time.
S: That's true. But you're not running true OS9, probably, because you're running on...
K: Yah, it's OS9.
S: Does the C compiler work the same on it.
K: Yup.
S: So for instance a lot of my C programs would work, ported over to that machine. So that might not be such a bad way to look at [upgrading OS9].
K: Especially since 386 machines are coming down real quickly.
S: Yah, I might just do that to have a PC portable.
K: That automatically gets you into a lot of markets, cuz everybody's got a 386 laying around these days, or will be.
S: Yah, that's true. So that's definately something to look at for the future then.
K: Yah, so, the question was, why didn't we do that, and...
S: That's a good question.
K: When I was first looking at OS-9000 it wasn't fully debugged at the time, and I still think it's not fully debugged now. Second, things were a little bit costly at the time. And because other than I didn't want to write the windowing system in C...
S: Much less hack it in `386 assembly...
K: Exactly. That was the last thing on my mind. Yah, we

all love the motorola stuff.
S: So the decision has been to do the windowing system in 68k, in 68k assembly. Which is the way to do it because that's the way OS9's done anyway.
K: Right.
S: And that's its big advantage over unix is that it doesn't have the overhead built in with C compiled programs.
K: As a matter of fact that's one of the things again against OS-9000, because it's written in C.
S: You're kidding!
K: Yah, 95% of it is written in C. File managers, everything.
S: So they essentially took OS9 and put it in C?
K: Exactly.
S: Oh man. So you're getting some of the benefits of the good structure internally of the operating system over unix, but you're running into the problem of still having the excess overhead built into all your C [software].
K: Of course, this depends on two things, one, how good is your C compiler.
S: Yah, well, see, now there's the thing. I did a lot of study on that back at Purdue. And the big problem with C compilers is that because they compile all the way down to object before they link, a lot of assumptions have to be made about every function. Every function might be recursive, therefore all variables have to be passed on the stack. Which means that say you just want to call a little function that plays around with your [passed value] and does a little calculation, and comes right back to you. Well, you've got to pass the variables on the stack every time [it's called], which is actually a good deal of overhead. I mean, you could have just as much overhead code in passing the stack as the actual code in the function may have. So what will end up happening is that if you're calling this function, let's say, one time for every character in a file you're processing. Multiply that by the overhead and you've easily got half overhead and half program.
K: Right. Of course if you go to a fast enough processor, like where we saw it up at BUSCON (VME standard bus convention) in Boston last August I think, or September, Microware had OS-9000 running there, and they haad it running on two machines. I think a 68020 and a 386.
S: So OS-9000 also runs on 68k?
K: It's written in C. It can be ported on anything you want. Now that's very handy. Running on a 386 or a 68030 it's not going to matter if it's running in C because it's going to be fast.
S: But the problem still is you've got to have that fast processor.
K: It smacks of the Intel way of doing things, which is to rely on faster hardware to take care of it.
S: Take care of your lack in software. But that's where we have an advantage back in 68k OS9 because we've [well, Microware had] done everything in assembly.
K: However, I refuse to dump on OS-9000, because in 4 years we may all push over to it.
S: I'd rather see it done in assembly first, but yah.
K: And somebody may eventually take the 386 part and optimize it.
S: Well, not only that, but somebody may take the C compiler apart and optimize it. I've actually got a design for that.
K: They also gave the guy who did it (OS-9000) pretty much free reign. Very similar to what they did with us on the upgrade. So he added in all this stuff like text editing on the command line, ala my editor, and a neat way of having the buffers shared between SCF and the serial driver. So they don't use separate buffers, they use the same interrupt driven buffer.
S: Do what now? [translation: could you please explain that]
K: Right now, SCF has it's own editing buffer right? The driver has it's own interrupt driven buffer to hold the [incoming] characters. But it still comunicates between SCF and the driver one character at a time. Which is a lot of overhead and memory usage. So what he did was if it's an interrupt driven driver you can set a bit to tell the SCF to share the buffer and where it's at. So like doing an I$READ of 50 bytes SCF can go straight to that buffer and pluck it out, update the pointer for the driver, and whosh, ya know.
S: Major speed increase there, especially if you've got a big file coming in on the serial line.
K: So it's (OS-9000) a neat thing. And the memory module directory is the neatest thing of all. We've wanted that for a long time.
S: How's that work?
K: You have a makmdir and a chmdir so you can do...
S: Do what? [translation: huh?]
K: Makmdir makes a module directory. Say you have several people on the system, they're all compiling modules that have the same name or something, it doesn't matter because you're in a different module directory.
S: Oh, so you can make your own execution directory like, only in memory.
K: Yah, exactly. So what you can do is a makmdir Scott, and only you can see those modules and only you can execute them.
S: That's very cool! Cuz that's always been a problem if you've got enough people working in the machine at the same time, and somebody's trying to compile something that somebody else is trying to use it... The guy compiling is wondering why the change he just made isn't showing up, and the guy using it goes what the [omitted] did this change for?
K: And when you do a mdir you don't see a whole lot of modules you don't want to see.
S: Right, that too. Like I don't need to know about all these screwy system modules if I'm just a user.
[also discussed here was that OS-9000 does not have the same disk format, this portion of tape was buggy]
K: Another question is why we didn't all move to the Amiga. At one time, I was all for it, to tell the truth. The problem was A: it wasn't readily available, B: certain things about the Amiga drive me nuts.
S: Like what?
K: Well, one of them is their bit banger disk driver.
S: You're kidding?
K: It's okay, but it's not a regular chip. I wish they used a regular, everyday, Western Digital chip like the Atari uses.
S: Oh, they didn't use a compatible chipset for the floppy controller.
K: They actually use a serial to parallel converter.
S: They bit shifted it out there.
K: Which means that when you format a disk, you actually give it all of the data.
S: Oh man.
K: Well, that has a slick idea because you can say, Oh well, I can be compatible with any disk format out there right? The problem was of course that they did this horribly slow because they had to do...
S: all the conversion by the processor.
K: And track read and write. If you want to write one sector, you have to read in the entire track, find and modify that sector, and then write the whole track back out.
S: Ugly! Oh man...
K: Ugly is right. They get around this because of the fact that they have DMA and they cache stuff up and things like that. But still I didn't like [it] and the worst problem was we did not have an OS9 compatible disk driver and I don't think there is one yet. So, obviously both OS-9000 and OS9 have the big problem of compatibility. You can't

transfer [omitted] over except by serial port. I'm having a great time now because I still use my editor on the CoCo, edit up the source code, stick it on the prototype, compile it, take the disk, stick it in the MM1, and run it. So I like being able to read and write CoCo or ST or MM1 disks, take them back and forth between the machines. I like that a lot. At least as a beginning. Sooner or later I will be sticking my CoCo in the closet.
S: At the point at which everything you need to do is on the MM1.
K: Exactly. With all the 68k Utilities, and the TOP stuff from Germany, I think we won't have a lack of software. As far as utilities go. We'll be pretty well taken care of.
S: It's the big applications that we'll be lacking at first.
K: Yah, and what [bugged?] everybody on the CoCo of course was a lack of space. We have no problems like that now.
S: That's true, we've got the memory and we've got the SCSI for mass disk storage.
K: And we've got the space for processes, because that's something a lot of people say, well, I could port this over, but I have to do some tricks to get around the 64k limit. And this way, you can edit a half meg file in memory. I think in many ways we're kind of chipping away at the damn that holds people back. And it's going to really help out. I feel a lot less [?] myself.
S: Now you don't have to be worried so much about memory constraints.
K: Exactly.
S: At the same time though, all the good experience in doing that has probably taught you some good programming skills.
K: Exactly. The difference on the CoCo was you would write something and look at first and foremost at the size standpoint.
S: And the problem is we were trying to play catch-up with all the other big machines who didn't have to.
K: Exactly. So we'll be in much better shape here. I feel kinda free myself.
S: I can quote you on that right?
K: Yah, you can quote me on that. A lot of people ask me that kind of question all the time - What's the difference between the two machines, Is it like going back to level 1? Well, it would be if you where on a 64k machine. But you're not, you're on a 1 meg machine. And those of us who have had 512k Atari's for a long time, we didn't feel constricted. And with one meg, you really don't feel constricted. Even with possible fragmentation of memory, it is so rare a problem, I've never run into it. It's a regular OS9, as you know. People are worried about that. Mostly, the commands have more options to them.
S: And they all have the dashes now, like should have.
K: Yah, and actually we need to have people write some coco compatible stuff like that so that people can get used to it. I'm sure you've done the same thing I do, you go back and type dir -e on the coco and it doesn't [work].
S: Yah, I keep doing that. In fact, the coco right now is just running the network, and I don't really use it, except it's got the modems on it. So unless I'm pulling stuff in and out, in which case the first thing I do is slap it on a coco 3-1/2 and read it into the Atari.
K: Of course, some things that we have modified for the CoCo over the years I miss. One in paticular is the way we modified the shell to execute scripts out of the command directory.
S: Those kinds of things we can re-write a new shell (for OSK).
K: Yah, we will have to re-write a new shell to get some of this stuff. But other than that, everybody's always worried, no, no, it's very easy.
S: Very easy to convert, yah.
K: It really is.
S: Yah, I've got an article I'm writing in the OSKer that is OSK for OS9ers, that covers the basic set of commands and what the differences are. Really, in the end it shows, hey, there's not that much different.
K: Hey, somebody a long time ago, maybe it was Frank [Hogg], uploaded a file to the OS9 forum on CIS a file that listed the help output of all the commands. Some stuff I've really taken to, like the dsave -er, execute and rewrite over files with the same name. I use that a lot when I'm copying stuff around.
S: Really, cuz I'm used to the dircopy.
K: Things that were a little bit difficult on the coco become a little bit easier here. And probably vice versa sometimes.
S: It's more a case of just getting used to the new options.
K: Yah, and of course in level 2 it's the same thing, everybody put help on all their commands. [the -?]
S: Yah, all the commands do that in OSK. In fact, I'm more peeved with the few commands that people have written for OSK already that don't. And what's more I get really irritated when I get on Unix and I put a -? on something I want to know the options to and the shell barks back at me that it can't find ?. It's like, argh! Now I gotta go pull up that stupid man page and it's going to give me all this [omitted] I don't need to know when all I want is the stupid options.
K: And of course the other thing we've done is stepped up with disk [cache] and SCSI DMA. And that really helps us.
S: Yah, so that the through-put even just on floppy is faster.
K: And of course the ghost modules is another big thing.
S: Now, I really haven't looked at ghost modules too closely. That's where it sticks into memory?
K: Now you notice a lot of the stuff that you do now, dir and copy...
S: Yah, the first time you run dir or copy it links itself into memory and it stays there until you unlink it.
K: It stays there until it needs the memory. Which is never.
S: Yah, if you've got a meg of memory, it never actually needs to unlink them.
K: I think only once on the Atari, I think the C compiler is also ghost modules. I would run a compile and I would have all sorts of stuff in memory, and then I need it for graphics screens, and that's when it took them off. I was suprised cuz it went and I was wondering why. Then I relized that it had given up that space. But that only happened to me once. I had a whole lot of screens open. Of course the other thing is we're going to need more disk space, which is [why] we've got these high density drives. When each picture is 64K long, twice the space of a CoCo picture. It's been a good training ground for us. I wonder if you could actually compare us to, have you ever read Frank Herbert's Dune?
S: Yah! Well, I've watched the movie. I haven't actually sat down and read the whole book yet.
K: All these kind of stories where these people are put on a really barren place. They get along there, and when they break out to another spot, they're lean and mean, ya know. And so we've had excellent training over these years of working in small places, doing the best we can with that, and optimizing for speed and everything else, while these other guys have always had this open space and not had to worry.
S: Yah, and they're programming techniques have been sloppier because of it.
K: Exactly. And so, I think, when you take our same techniques that we have done and move the move to the 68000 we can really, really, speed.
S: And outperform everything all the other stuff people are

doing. We've got the advantage now. It's just a matter of time before we're actually built them all up.
K: And that the other thing, people say it takes years for these things to happen. It really does, these big companies like IBM, these people have millions of dollars of backing, 20 people working on a project. Of course they can come up with something, ours just takes longer. As I mentioned on the OS9 forum the other day the wierd thing about OS9ers is they were patient. And if we see something new, we chew it over for a couple of years. Will it work out for the next 50 years. Can I still be running this in the year 2000? A lot of us are still running utilities that were written in 1980.
S: Yah, like the edit command for 6809. If it wasn't for uMacs I would have done something to get that on 68k. Because that was my editor. I finally learned uMacs to the point where I like it better now.
K: One of the things that's wierd about OS9 is we're frugal about stuff and we don't like to write the same thing twice. Not ever. Other people are into instant gratification. Which I can understand. Like the PC'ers, who immediately bypass the bios and go straight to screen memory. Which has caused them no end of trouble over the years.
S: Yah, you talk to the screen directly and you get faster display, but you don't have [operating system] control when you try to go multi-user later.
K: How do you change thousands of programs that are all competing against each other to be faster than the other guy.
S: Yah, the bottom line is talking to the hardware is always the fastest. But if you don't have that layer, you can't switch off programs.
K: See, the amiga guys are in the same kind of position. Most of their programs, so they tell me, go directly to the screen hardware. We used to argue with the RSDOS'ers as a matter of fact, they'd say, this program, the same program under OS9 runs slower. They'd say, will somebody buy it? We'd say, yes, because people are more patient. We realize the hardware's going to catch up. The hardware's going to get really cool one of these days, and if my stuff still works.. Of course now you're going to tell me to implement CoCo escape codes, right? But in the main, our stuff is written to be device independant. We like that because the hardware will always get better. And you can see it even now with this stuff. When we move to the 68000 it's a lot faster. And Kevin Pease has a 68030 unit running the same windows here, his stuff just SCREAMS. So as time goes by, we're going to benefit from our device independance. So people into instant gratification, people who have to run this canned program now, this game now, that's fine. Use your PC for that. Use your Atari, use your Commodore, use whatever you've got. And I have no beef with those people.
S: Yah, it's like right now I'm using a desktop publisher on the PC. Because it's the only one available that's decent enough and will talk to a laser. But I keep thinking in the back of my head, what I can do to write one on an OS9 system.
K: And now we actually have the freedom, the ways to do it. Of course the other main thing is OS9 has been, at last, [for?] the hobbiest. We may even have a PC around, like you do, to do something specialized like that. But you want to have fun, or you want to learn, you use the CoCo or anything running OS9. A lot of people on the board use pc's at work, but when they come home, they learn and play in their OS9. It's a learning experience. I know people who have started in OS9, when they got their coco the first they had was OS9. They didn't even use RSDOS by the way.
S: Yah, I actually started in RSDOS and then was doing some very OS9-ish things like redirecting i/o by playing around in assembly language with the roms in the CoCo. And I was in the midst of that project, had most of it completed, when I found OS9. It's like, oh, shoot, they've already done all this.
K: Oh, yah, I was the same way. I like a lot of people had heavily modified the rom. Trouble was, other people couldn't use my stuff. When I use OS9, I said, gee, now I write something I can share with other people and they can add it in, and it will work with everything else. Ya know, WOW.
S: And, it was multi-tasking already. So it had some neat features we hadn't gotten to.
K: So these people that start with OS9, and then, say for instance, move to MSDOS, they have far fewer problems because they're used to a disk operating system. So the glitz and the glamor, some people say our programs don't have the flair, they don't have the fancy colors on the screen and stuff like that. Well, yah, but thats window dressing. Most OS9 heavy duty software works great! And it's very [proficient?]. It may not have pretty little windows, but it works. And it works solidly is the main thing. I am always astonished with some of these forums with people talking about how their programs crash all the time. And I go how can you put up with it? I'm [using?] a multitasking machine and a multitasking machine is useless if it crashes all the time.
S: Even if there's just one program going astray it'll take the whole machine down with it.
K: Sure, so we do have bugs, but in the main, if you have a program, it's going to work. And if it's fouled up it may foul up it's own self, or it may foul up it's own files, but it's very rare to find an OS9 program that actually messes up the machine.
S: Well, I've actually written one that does, but that's the exception rather than the rule.
K: For instance, if you wrote a Basic09 program, unless you're doing [?] or a bad stat call, you're not crashing the machine because it's a solid program to start with. On the whole we're far better off than most people. A lot of us like me leave our machines up for 24 hours a day for months.
S: In fact, I just retired a machine that has been running practically 24 hours a day for the past 6 years.
K: I think the right incentive to keep us with it will keep us going for a long time. As the hardware gets better, we will reap the benifits of it. But mostly OS9ers are into learning stuff.
S: To get on to other things of minor interest, how did you first get started into computing?
K: Wow.
S: Big question.
K: Back when I was 11 years old, 1964, my parents gave me an analog computer kit. In 1971 I was at UNC chappel hill, taking programming courses there in PL/1. When everyone thought that PL/1 was the language to end all languages. That was when they were first starting out UUCP at North Carolina. Between Carolina and Duke was the first UUCP.
S: Really?
K: Yah, that's where they started using that. I wish I had been involved in that but I wasn't. I was taking calculus courses that used the computer. Then I joined the army, and unbeknownst to myself I was using incredible technology. We were using 32 bit RISC processors [in?] these jammers we had and I didn't know it. But in 1977 a friend of mine introduced me to Byte magazine. So when I got out in '78 I went down to school at [?] state and [studied in?] electronics engineering. In '79 I [built?] a 6800 processor from scratch.
[the tape is reproducing static very clearly here]
K: I was running a 6800 system with a VDG chip I finally got one k of programming memory, 4k of graphics memory. [Was?] using a scope for output, vector display. 32 by 32 dot

screen and managed to get up to 256 by 256 vector using d to a converters. Kinda slick because I had stored in memory a picture of a transitor curcuit, and so what happened is i hooked it up to the scope, people would come over here and they'd say lemme see your scope. I'd turn on the power and it came up with a picture of a transistor curcuit on the scope. I built a voice synthesizer on it, had voice input on it, a math chip on it, it was neat. When you built something, it was new. You could build something that nobody else had.
S: Very easily.
K: And nowadays of course you either buy chip to do it or a machine already [?...]. You kinda miss the old days. So when the CoCo came out, I went to work for Radio Shack just so I could get a discount on the machine. Here it was with a 6809 and a VDG chip.
S: And you already knew the machine.
K: And I wanted to move to a 6809, so I said, aw man, I gotta have it. As it turned out it took me a year after they came out with it before I could actually afford one. And Marsha [my wife] gave it to me for christmas, and I think she's regretted it to this very day. I still worked at Radio Shack when I bought my first floppy drive for $450 on sale. Bought one of those graphics tablets for $400. Some friends of mine at the club here got into OS9, came to me after messing with it, and once I got into it I was very excited. I guess I got into it about '83 or '84. Whenever it came out. I remember I was at Ft. Worth Rainbowfest when it came out.
S: That was '84 from my notes with Frank.
K: Okay, so I was there when it came out. As a matter of fact we had been on a cross country trip on the way stopping at every Radio Shack, and asking them [if they had] OS9. And this is what really kills me. I stopped in a Radio Shack center in Dallas, Texas, and asked the guy if he had OS9 and he said yes. The problem was I didn't have the money to buy it. Because that was two days before the Rainbowfest, when they officially started selling it, if I had had the money I could have been the first official owner of OS9 on the CoCo which I thought was cool. I blew it. So they had about 50 or 100 [copies of] OS9 at the fest, and they were sold out just like that. They were gone. And like most people I looked at the manual and said, oh, [omitted] this is complicated. I stayed away from OS9 actually for a while after that because it did seem complicated. It wasn't until later that I realized how [quick?] everything was. And so about the next year, 85, 86 or something, I expanded my CoCo to 128K. Just before everybody else started coming out with the banker and all that. I went ahead and designed it and built it here, actually one for a friend of mine too. So we had 128K coco's here, and we decided what can we do with it? So I wrote a ramdisk for OS9 to use it. The company DSL came out with a 128K kit, and they called me up and I they started selling my ramdisk. I modified it for [their] board. So that was my first [?] product. And what I didn't know was that had made me known to people. I was not into communications, I wasn't on CIS, I had no idea. I sold enough of them it paid for my first hard disk which was a 5 megger cost me $800. That really hurts. People complain about the cost of drives, I just want to smack them. I went to my first RainbowFest at Princeton, and I walked in there and people would walk up to me, spot my name tag and say ``you're the guy who wrote the ramdisk''. I really had no idea. And that's how I met [? and ?] from LRTech, they said they had taken apart [my ramdisk] and used it for writting their hard disk driver. Which is funny because they way I had writen the ram disk was from a hard disk article in [?] Micro Computing, that was written by Steve Childres, who was Pete Lyal's boss.
S: So you taking that it sort of snow-balled...
K: into everything else. And then we all learned more as we went along. Later on I got up on CIS, went on to become the assistant sysop. About the same time, in '87, I build a 68008 co-processor for my CoCo 1. I used it to do the graphics. That design (the arbitration between the 6809 and 68008) is now implemented in the TC9, unless he changed that. It took me forever to figure out. So after writing that ramdisk and seeing how many other people were into OS9 that really got me interested. After that I wrote the Inside Level 2 book. Various people snuck me copies of Level 2 early. So I actually had all these notes written up for myself, and Frank said why don't you write something up on Level 2. Just a little pamphlet should help. As it turned out I had all these notes laying around. I thought it would be a good idea for Frank to print all my notes for me in a nice format. That's why I kept tossing stuff in there because I use it for my own reference. That [ramdisk] and the reference are the two things out there. But I met a lot of people that way. That's my favorite thing about OS9 is all the people. I love it.
S: Well, for one thing you do a something really neat that a lot of people like and you become instantly famous.
K: Yah, it's the rush is the thing. I told Mike Haaland [author of MVCanvas] wait till you go to the fest, these people are going to know who you are [and he didn't believe me]. Sure enough his first day, people went ``you're Mike Haaland, you're Mike Haaland''. They even asked him for his autograph. Now this is as embarrasing as you can get. Don't print that, but it's embarrasing. You know, I'm just you...
S: ``I'm just an average Joe, I just happen to have written this thing, ya know. Anybody could have done it.''
K: And that's what my biggest problem is, most anybody could write this stuff...
S: If they sat down and did it.
K: Yah, and my advantage has been I have had 24 hours a day for the past 4 years to do nothing but OS9.
S: Now explain that. How did that come about.
K: Marsha has her own business. And bless her heart, she's always had faith that OS9 will pay off. Eventually. And then I'll be able to take care of her. So actually, the world owes Marsha. Everybody else always says, well, how do you know so much.
S: It's the time you spend at it.
K: Sure, yah. If you had 24 hours a day, to play with OS9 and mess with it, you would know all this stuff too. I'm just like anybody else, [but] I just happen to have the time.
S: Yah, that's where I've had an advantage being a student. When my parents supported me I could fiddle with OS9 in all my spare time. [And now that I work for myself, it's almost all I do.]
K: The other great thing is that people have donated equipment too. Like Bill Brady he donated the Atari ST so that I could write OSK stuff, and Bob Santy bought me a 20 meg hard disk so that I could write windows for him. And Wayne Day gave me a CoCo3, because see, I couldn't afford any of this. I am more poor than the poorest person you know.
S: See, now that's something a lot of people wouldn't know.
K: So when people say, why couldn't you have done this or the other, I have to say because I don't have the money or resources. [He mentions about here that he can't even afford mailing letters out really.] Various people have donated various things. A lot of the stuff in here. I want to pay these people back one of these days.
S: Well, I tell ya, you are really paying them back with the work you do.
[the tape is barely intelligible at this point and bits and

pieces are missing]
K: The funny thing was, about six months before that [message on CIS about 1 Meg upgrade working], I was about to upload a file giving all of the details I could for guys like you. If somebody does this, send me one. Here's how you do it, and I will do the software. Tony had called me up, and said ``I'm really bored, I don't have anything to work on''. And I said, I'm about to upload a file on how to make a 1 Meg upgrade.
S: Talk about timing. [...]
K: So the other thing I do is I always give out my name and number at all my seminars, at the fests. So my number is out all across the country, and the interesting thing is I get the most unique phone calls from people, and I love that. Except for when my phone's off the hook when I've got to get some work done. I get up to a dozen calls a night, and several of them will be from people who I have never talked to before. [...] or they'll be somebody who's out in the middle of nowhere, never been near a club, there's nobody else that uses OS9 [...] and I have the greatest admiration for these people. These people have perservered under great odds. I kinda learned it [OS9] on my own too, but once you get on a major network like CIS or Delphi or something like that, and you've got all this help...
[the one hour micro-cassette runs out and I start another]
K: The main thing is, I like the people the most. I know some ham types who run OS9, just so they can have a BBS to talk to one another. Just for their friend across the city. And these people taught themselves, they've had no problem. They didn't start of with RSDOS, so they weren't confused. They thought this is the way computers work.
S: Which is right, in a way...
K: So to them, to move from OS9 to RSDOS would actually be as bad as an experience as these guys who move from RSDOS to OS9. Probably worse. So I wish I had more time. These days, I have to keep my phone off the hook a lot because I just have to get this work done. I kinda miss talking to all these people. On the OS9 forum the unofficial motto has been `there is no such thing as a stupid question'. The only dumb OS9 question is the one you didn't ask. Newcomers will often come up and say, ``I have a dumb question''.
S: Those are the smartest questions of all.
K: If you didn't ask it, you were dumb. Because you're beating yourself against a wall for no reason. So I love the way people help out. I love the way that OS9 people most of the time feel very strongly about handing out free software. Obviously it's gotten to the point where that has to change a little bit. MVCanvas, stuff like that. People wouldn't finish a project unless they get [paid back?] on it. So we're now moving to a new era here, of major commercial appliations and stuff. I think it's past time for it. I reserve my greatest admiration for somebody who comes up and says `remember that problem I had 3 days ago, I kept at it and I finally found the answer'. To me, those are the best people around. Because they did not give up. Somebody who does give up right away, I'm sorry, ya know. No loss.

Kevin and I could have talked on most of the night, er, morning, but he's got the windows to write and I've got the OSKer to put out. We agreed however, that ideas are not something that CoCo and OS9 people are short of. In that vein, I'm offering to serve as a buffer for ideas about Kevin's new windows. He says he's got a lot of the core routines done, and promises that the first release (with the new machines, MM1 and TC70) will be usable although somewhat minimal. Fancier stuff will be updated as we go along. But he is in need of input as to what capabilities people think the windows should have. The major questions of how exactly it is going to work have not all been answered as yet. But I think he's a little reluctant to outright ask for opinions, because he knows he'll get plenty of them. To cut down on Kevin's overhead, you can forward your opinions to the OSKer P.O. Box (see inside front cover) or e-mail to any of my network adresses. A complete list of suggestions will be compiled and forwarded to Kevin. And of course we at the OSKer will keep you updated as to the progress of the Darling windows. Hmm. Hey, Kev, what do you want to call this thing anyways?

FILE DESCRIPTOR: OSK_for_OS9ers
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, OSK on board
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS, StG@hummer.iupui.edu

I've had several people ask me what the difference is between OS9 and OSK. Well, quite frankly, there isn't much at all. I mean, all the commands (well, almost) you have in OS9 work pretty much the same way (well, sort-of), and programming under it is pretty much the same (ugh, maybe not).

If that's confused you already, then let me try to explain myself. If you know OS9, you know OSK. But, you have to get used to a few things that they've changed. First, every single time (except tmode/xmode) that they forgot to put the dash on options (like dir e) in OS9, they did correctly in OSK (like dir -e). Plus, they put -? for help on every single command! So if you forget an option, or want to see what new options they added with OSK, you can just ask the command.

For example, if you enter `dir -?', you get:

```
Syntax: dir [<opts>] {<dir names> [<opts>]}
Function: display directory contents
Options:
     -a          show all files
     -d          show directories with a slash
     -e          extended dir listing
     -n          treat dirs like files
     -r          recursive dir listings
     -r=<num>    recursive dir listing to depth <num>
     -s          unsorted dir listing
     -u          unformatted listing
     -x          directory is execution dir
     -z          read dir names from stdin
```

So, the dir with just e and x options has grown to include many more, but don't forget to put the - in front! Also added is * and ? wildcards in the shell, so you can do things like `list *.doc'. Plus, they've added a mode to copy. If you do a copy /dd/fromdir/* -w=/dd/todir it will copy all the files in one directory to another. The w option names what directory to copy the files into, and has to be used when copying more than one file with the wildcards.

Also new is the addition of shell environment variables. The first time you'll find these necessary is if you try to use uMacs, or any other program that uses the termcap database (for interacting with any type of terminal). For example, when editing files from my PC across a serial port to the ST, I set the environment variable TERM to `ansi' with the command `setenv TERM ansi'. What this does is create a storage area called TERM and put in it `ansi'. Then when I run uMacs, it asks for the TERM, and looks up

ansi in the /dd/sys/termcap file to figure out how to talk to it. Also, login sets a lot of variables when you log in. The printenv command shows for me:

```
HOME=/dd/USER/StG
SHELL=shell
USER=StG
PROMPT=OSK:
TERM=ansi
```

Oh, I almost forgot, you can set the shell prompt too. And if I do a `chd' it automatically goes to my home directory from the HOME variable, so I don't have to remember where my directory is. The environment variables are actually implemented separately from OSK itself (they can be done in OS9 the same way!). There are special calls and code written into the argument processing in C that pass environment variables when commands are forked. Which means that only C programs (most, if not all, of the commands in OSK are written in C) can do argument processing, unless code is added to the other languages (does Basic do this?).

And the greatest thing since sliced donuts is the addition of named pipes! The /pipe device now becomes a directory, into which you can copy files. I haven't actually used it much, but anytime you need to use a quick temporary file you can put in in /pipe for faster processing. But watch out, it's not like a ramdisk. Once you read the file back out, it's gone. It's just like a pipe, only you can store it with a name until needed.

The OSK C compiler is practically 100% compatible with the OS9 one. All your C code written for OS9 will covert with very little if any change. At the same time, they made some big improvements. The int declaration is now 4 bytes, and the compiler has helpful warning messages. Also, they implemented the unsigned char type! This has always been a pain as it was missing in OS9.

Basic under OSK is also compatible with Basic09. I don't know a whole lot about this (gimme a break, I don't use it), so somebody please find out and fill me in, but I have heard that certain variable sizes are different under OSK, and this could possibly mess up GET/PUT random access operations? I'll let ya'll know as soon as I find out.

Got any questions? Ask the Doctor...

FILE DESCRIPTOR: Origins_of_OSKer
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, Set Statable
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS, StG@hummer.iupui.edu

Why do we call this magazine the OSKer? Because it was the only name that fit I suppose. But I didn't invent it. Really! Look in the May/June '88 issue of the MOTD, page 3, middle column, 2nd paragraph. ``You Atari OSKers and CoCo Folks both can pitch in...''. So don't blame me.

I have started a few conventions though. For one, spelling it without the apostrophe between OSK and `er, as one would tend to. This was suggested by Alan Sheltra (Sysop@Zog) I believe. Although I originally didn't take to the idea, I caught on. It's easier to pronounce correctly for one thing. Which brings me to the second convention - OSKer is said `oscar', just like the fish.

The cover artwork, also done by Alan, was derived from a suggestion by Frank Hogg. He wanted to see a more amiable cover, depicting the two camps working together to `save' users from the sinking CoCo (we had originally planned him and Paul in a ring duking it out). I just hope no CoCo-Nuts were offended by it...

It was a couple of months ago when members of what is left of the Indy CoCo Club (which we really ought to rename the OS9 club) were discussing where the Rainbow is going (can you say pishtochk?), and that we really needed a good magazine just for OS9. Again and again we came to the conclusion that somebody should start one. It was several weeks later before it occured to me that if somebody didn't get off his brain (I was standing on my head at the time) and get the ball rolling, nobody else would.

So I bounced some suggestions around the StG network for possible titles, content, and so forth. What I got back was nothing but positive response, and plenty of it. It seems that just about everybody had been thinking the same thing. That I should get off my head start working on it.

So, with a few hundred thousand keystrokes, a bit of my own money, and the interest of a large number of people in the OS9 community, as well as the assistance of several others, the OSKer has been born.

Special thanks go to Alan for his myriad suggestions, collecting a list of people for the first issue, and drawing the cover and ad copies. Additonal thanks to Chris (the Bug) for the pop utility. Chris is already working on about five more programs, so we expect him to be a regular. And I can't forget Kevin Darling letting me talk to him at length, or Frank and Paul for their help too.

But most of all, thanks to you, the reader, for taking the time to show an interest in OS9. With your help, we can get together and benifit from each other's knowledge needs, and effort.

Hey, if that isn't corny enough for you, I've got a ten pound bag of nacho's... anybody got some cheese? Please!

``The Atlanta CoCo Fest''

Oct. 6-7, Northlake Holiday Inn, Atlanta Georgia

CoCoPro handling ticket sales, room reservations. Tickets are $10 for one day, $15 full show. Rooms are $49 plus tax, and the first 125 people who purchase hotel rooms through CoCoPro receive free one night pass for each room/night.

IMS, Microcom, Zebra, B&B, and many others will have booths. Kevin Darling, Art Flexser, Dan Robbins, J.D. Walker are doing seminars.

For more info call Dave Myers at CoCoPro, (313) 481-3283

The OSKer Software List

This regular section is a list of all software that is done and working, being written, or just wanted. The list is kept alphabetically for easy lookup, and to the right will be the status of that program. A 9 or K indicates that the program is only available (being written, wanted for) OS9 or OSK specifically - otherwise both are assumed. The entire list of codes are as follows:

9 - for OS9 only
K - for OSK only
W - Wanted, dead or alive...
C - Code is being written
A - in ALPHA test
B - in BETA test
D - Program is DONE and ready for purchase
$ - Program is being sold
S - Program is being sold as Share-Ware
P - Program is in Public-Domain

To request a Wanted program, mail a short description to the OSKer (P.O. Box 24285 Speedway IN 46224), call us up (317-241-6401), or post it to the OSKer news area on StG-net. The most asked for programs will be featured in detail.

If you are developing a program, please let us know about it. Not only is the general public eager to know what is going to be available soon, but it may help someone else decide against creating a similar product.

If you have a program available, whether for sale, share-ware, or public-domain, mail a copy of the program to the OSKer. All programs submitted for verification will also be considered for reviews, and can be returned. Programs certified by the Rainbow will be added by request.

Because this is the first issue, there is only a couple of program in the list. Hey, sorry - check back next month!

Feature Wanted Program

Frank Hogg has suggested a program that would serve an interface (maybe with pull-down menus, windows) to the CIS, Delphi, Genie, etc. The purpose of which is to make it easier to switch between different services, as all the basic functions would be handled by the program. Sounds like a really neat idea - Thanks Frank!

The OSKer Software List

Program	Description	CODE
DTP	Desk-Top Publisher	KW
DB9	OS9 Data Base	C
Term	Terminal Prog, Common Interface to Svcs	W
MVCanvas	Graphics Paint (Editor)	9$
Spr Sleuth	68000,008,010 Disassembler	K$

FILE DESCRIPTOR: uMacs_Primer
OWNER: Scott Griepentrog
ATTRIBUTES: Editor, OS9 Freak, uMacs Hopeful
ALLOCATION MAP: Sysop@Root (StG-Net), 72427,335@CIS, StG@hummer.iupui.edu

OSK comes packaged with a ready-to-go full screen editor called uMacs. The name is derived from Micro-Emacs, which is derived from Unix's Emacs, which I believe is derived from something like Editor with Macros. It has been in development on Unix for years (mostly by hackers), is quite large, and even this smaller version we have can be a bit too much for the average user to get used to.

uMacs uses the termcap routines (also from Unix), which allow a program to work with almost any terminal. This makes it usable from an serial line as easily as from the console display. Shove any dumb terminal with cursor addressing next to your OSK machine and you've got an editing station. You can even use a PC as a terminal, as I am now.

Other features of uMacs are multiple (windowing) edit buffers, a C source mode for auto-indention, and the capability to redefine control keys for different functions. There is a way to save your prefered key settings, but Microware conviently forgot to include how to do this in their documentation.

So uMacs is a pretty powerful editor - but why are people afraid to try it? After one look at the massive amount of control sequences one would seem to have to learn to make use of it, I balked too. But out of desparation for a good editor in OSK I forced myself to learn.

I've been using the old EDIT command from 6809 OS9 for what seems like an eternity. Hey, it works from any terminal, has global change and macro writing features I still like much better than any screen editor's, and it doesn't take a lot of memory. But in OSK they put only the stupidest commands into `EDT' - much like the Basic09 editor (which was oddly enough the major factor in my giving up use of Basic09). But enough of my problems...

To make it easier for the rest of you, I've created a list of the more commonly used uMacs control sequences and what they do. It's separated into sections based on the different type of sections. You can use this as a handy `cheat sheet' for the basic editing functions.

Of course, ^X means to hold the control key down and press X. But uMacs also uses the escape key to trigger certain sequences. For example, to exit uMacs you press ESC, then Z. This is shown in the form M-Z, where the M- means the escape. Don't ask me who came up with the idea to use M- for escape - if I knew I'd be giving him what for too.

Another odd thing is that you can preface commands with ESC, key in a number, and then the control sequence for the wanted function. Most commands will repeat that number of times, others (like goto line) will use that as a numeric argument. Also, ^U can be used to set a repeat count, and ^G is used to abort most any function. And if uMacs gives you the message [Key not bound], what it's saying is it doesn't understand the key sequence you hit.

MOVING AROUND

begining-of-line	^A
end-of-line	^E

forward-character	^F	
backward-character	^B	
begining-of-file	M-<	
end-of-file	M->	
next-line	^N	
previous-line	^P	
next-page	^V	
previous-page	^Z or M-V	
next-word	M-F	
previous-word	M-B	
next-paragraph	M-N	
previous-paragraph	M-P	
goto-line	M-G	(press ESC, then line #, then ESC G)

MAKING SPACE

newline	^M	
newline-and-indent	^J	
insert-space	^C	
open-line	^O	
handle-tab	^I	(plain tab, except TMODE controls spacing)

OTHER STUFF

buffer-position	^X=	
clear-and-redraw	^L	
help	M-?	(goes split screen - ^X^O to swap)
exit-emacs	^X^C	(aborts changes)
quick-exit	M-Z	(save all changes and exit)
i-shell	^XC	(temporary shell)
quote-character	M-Q	(puts actual next char in file)
redraw-display	M-^L	(redraw display with cursor at center)
	M-!	

KILL AND YANK

kill-to-end-of-line	^K	(puts text in yank buffer)
yank	^Y	(yanks text back in)

DELETING

delete-previous-character	^H	
	^?	(this is the DEL key on PC's)
delete-previous-word	M-^H	
	M-^?	
delete-next-character	^D	
delete-next-word	M-D	

FORMATTING PARAGRAPHS

fill-paragraph	M-O	(this reformats a paragraph to fit width)
set-fill-column	^XF	(set width - ESC (#cols) ^X F)

EDITING ANOTHER FILE

find-file	^X^F	(load up a new file without exiting)
next-buffer	^XX	(get next file on command line or found)
insert-file	^X^I	(load file into current)
save-file	^X^S	(save current file)
	^XS	

SEARCH AND REPLACE

search-forward	M-S	(search for string)
search-reverse	M-R	
query-replace-string	M-^R	(replace with prompting)
replace-string	^R	(replace, no prompting)

Apologies to our readers for not being able to print the article 'Playing Chess in C' in our first issue. We simply ran out of space, and it's too big. Maybe if Kevin didn't have so much to say... We will print it in our next issue, along with more suprises and fun. And just maybe I'll get the hang of this stupid DTP, or get irritated enough with it to start writing my own. Anyways, thanks for your support! We can't do it without you. Let us know what kind of articles you want, and get your submissions in to get your OSKer for free! - StG

I Want My OS9!